LONTAR KOMPUTER VOL. 9, NO. 3 DECEMBER 2018 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2018.v09.i03.p02 **e-ISSN 2541-5832**
Accredited B by RISTEKDIKTI Decree No. 51/E/KPT/2017

Implementation of Parallel Processing on Multi-Object Recognition Software

Midriem Mirdanies

Research Center for Electrical Power and Mechatronics, Indonesian Institute of Sciences (LIPI)
Komp LIPI Bandung, Jl. Sangkuriang, Gd. 20. Lt. 2, Bandung 40135, Indonesia
midr001@lipi.go.id

Abstract

Multi-object recognition software on Remote Controlled Weapon Station (RCWS) had been implemented in previous paper using Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) methods, but the processing time in one cycle is quite slow so it is need to be optimized using parallel processing. In this paper, implementation of parallel processing on multi-object recognition software has been done on a multicore processor. The Openmp Application Programming Interface (API), C programming language, and Visual studio Integrated Development Environment (IDE) is used to implement the parallel processing in this paper. The parallel processing was implemented in the for loop of the matching process between the capturing object from the camera and the database under two conditions, i.e., the original of the for loop syntax and after optimization of the for loop syntax. Experiments have been done on the core processor i7-4790 @ 3.60Ghz, 8 GB DDR3 of memory, Windows 8.1 OS using two, four, six, and eight cores to recognize one, two, three and four objects at once using SIFT and SURF methods. Based on the experiments, it was found that the processing time in parallel is faster than sequential process, where the fastest of the processing time is obtained after optimization in the loop syntax, with the processing time in recognizing one to four objects using SIFT method is 927.13 ms (8 core), 1019.31 ms (6 core), 1190.72 ms (8 core), and 1283.05 ms (4 core), where the sequential processing time in recognizing one to four objects is 1067.35 ms, 1164.78 ms, 1352.93 ms, and 1497.35 ms, while the processing time in recognizing one to four objects using SURF method is 1157.13 ms (8 core), 1517.83 ms (6 core), 1572.14 ms (4 core), and 1472.64 ms (6 core), where the sequential processing time in recognizing one to four objects is 5635.99 ms, 6268.47 ms, 3256.63 ms, dan 3883.78 ms.

Keywords: *Parallel processing, Multicore, Object recognition, RCWS, C language*

1. Introduction

The image processing applications are the application that requires a high specification computer or parallel processing techniques to speed up the processing time, especially in applications that use a complex algorithms or methods. Some publication about image processing have been reported, Park et al. have implemented the direct calculation of inter-particle distance in suspension by image processing using monte carlo method [1], Saleem et al. explain a comparison of feature points method on multisensor images [2], Husin et al. [3] on the poisonous shrimp detection system for litopenaeus vannamei using k-Nearest Neighbor (k-NN) method, and Mirdanies et al. [4] has also successfully implemented the multi-object object recognition software on Remote Controlled Weapon Station (RCWS) using Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF) methods.

Particularly in the publication of mirdanies et al. [4], the application program created has been divided into three parts i.e. reading data from kinect and simulating the results, object recognition process, and data transfer to the ballistic computer, where each part communicate using shared memory. This technique is effective to speed up the process and avoiding any collision or delay, because it is not necessary to wait the other unfinished processes. However, the object recognition process is still quite slow because the process runs online or real-time to match many data at once. Based on this, it is necessary to optimize the object recognition process using parallel processing techniques.

Parallel processing can be performed on multiprocessor or multicore using Distributed Memory Processing (DMP) or Shared Memory Processing (SMP). The Application Programming Interface (API) that can be used is Message Passing Interface (MPI) and openmp [5][6][7]. Several studies of parallel processing of DMP type have been done by Pinho that implements object-orientation in distributed-memory parallelism called Object-Oriented Parallel Programming (OOPP) [8], and Oger had done parallel processing using distributed memory parallelization technique on Smoothed Particle Hydrodynamics (SPH) [9]. The research on parallel processing with the SMP type has also been done by Phillips that implements classification algorithms on remote sensing (multispectral) [10], and Amritsar on dense particulate system simulations with computational fluid dynamic (CFD) using openmp [11].

Research on parallel processing by utilizing multicore processors has also been done by Mirdanies using QtConcurrent API with Integrated Development Environment (IDE) using Qt Creator, where the method was used is divided a complex process into two new sub-processes, and each process runs on a different thread [12].

In this paper, a multi-object recognition software had been optimized [4] using SMP type of the parallel processing on multicore processors. The method and API used in this paper are different from the previous article [12]. The Parallelization method is done in the loop process and API is using openmp. In addition, the programming language was used is c language with Visual studio IDE. The experiments have been done on two, four, six, and eight processor cores using SIFT and SURF methods to see the processing time on the i7-4790 core processor [13].

2. Research Methods

Diagram of multi-object recognition software in this paper can be seen in Figure 1.

Figure 1. Diagram of the multi-object recognition software

Figure 1 shows a diagram of a multi-object recognition software, where the camera was used in this paper is a c720p logitech camera which mounted on a gun barrel, it can be seen in Figure 2.

Figure 2. Logitech c720p camera

The Central Processing Unit (CPU) was used in this research is HP Pavilion with specifications: i7-4790 @ 3.60 GHz core processor, 8GB DDR3, NVidia GeForce GTX 745, 1T HDD, and windows 8.1. The processor was used in this study has four cores with the number of threads is eight thread that can be seen in Figure 3.

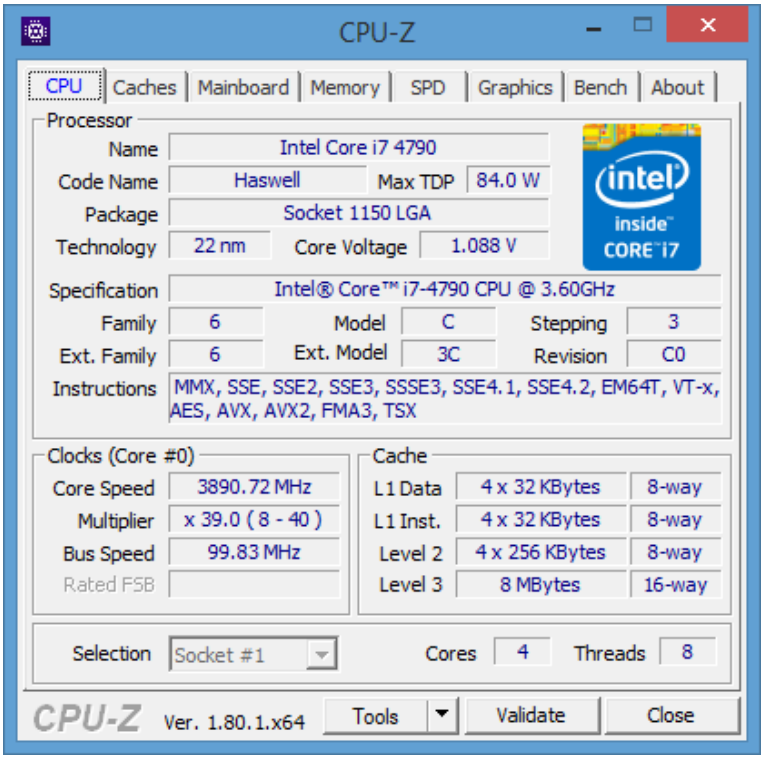


Figure 3. Specification of intel i7-4790 processor using CPU-Z [14]

The multi-object recognition software using SIFT and SURF methods that have been developed can be seen in Figure 4.

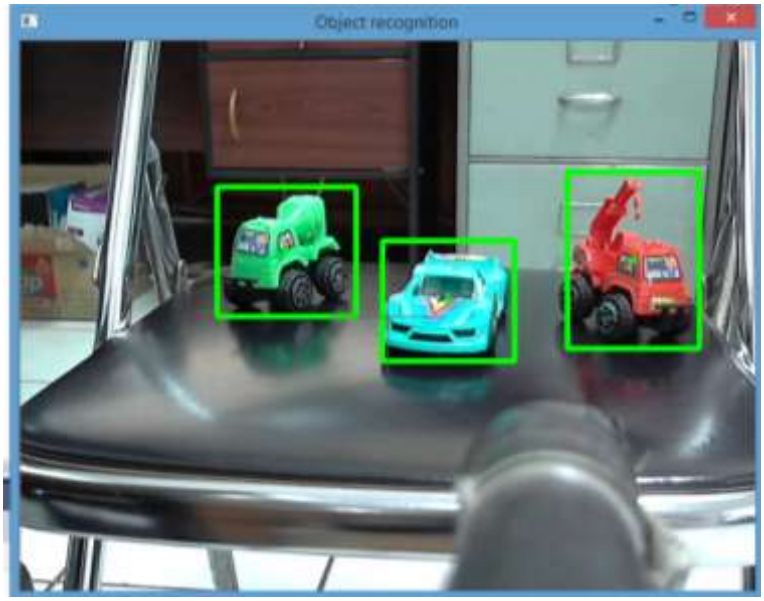


Figure 4. Multi-object recognition software

Figure 4 shows the display of the multi-object recognition software to recognize three objects at once. The objects were used in this paper are seven toys, i.e. wrecker, concrete mixer, blue sedan, green sedan, white sedan, wheel loader, and motorcycle. The display of the seven objects used can be seen in Figure 5.

Figure 5. The objects were used in the paper: (a) wrecker; (b) concrete mixer; (c) blue sedan; (d) green sedan; (e) white sedan; (f) wheel loader; and (g) motorcycle

The display of each object from several sides and different distances is stored in the ".yml" file, with the amount of data in each object is 50 data. The details of the objects can be seen in Table 1.

Table 1. Objects database

No	File name	Object	Method	The amount of data			
				Object name	Images	Keypoints	Descriptors
1	sift_mobil_derek.yml	Wrecker	SIFT	1	50	50	50
2	surf_mobil_derek.yml	Wrecker	SURF	1	50	50	50
3	sift_mobil_molen.yml	Concrete mixer	SIFT	1	50	50	50
4	surf_mobil_molen.yml	Concrete mixer	SURF	1	50	50	50
5	sift_mobil_sedan_biru.yml	a blue sedan	SIFT	1	50	50	50
6	surf_mobil_sedan_biru.yml	a blue sedan	SURF	1	50	50	50
7	sift_mobil_sedan_hijau.yml	a green sedan	SIFT	1	50	50	50
8	surf_mobil_sedan_hijau.yml	a green sedan	SURF	1	50	50	50
9	sift_mobil_sedan_putih.yml	a white sedan	SIFT	1	50	50	50
10	surf_mobil_sedan_putih.yml	a white sedan	SURF	1	50	50	50
11	sift_mobil_sekop.yml	Wheel loader	SIFT	1	50	50	50
12	surf_mobil_sekop.yml	Wheel loader	SURF	1	50	50	50
13	sift_motor_racing.yml	Motorcycle	SIFT	1	50	50	50
14	surf_motor_racing.yml	Motorcycle	SURF	1	50	50	50

Table 1 shows the file name with the ".yml" extension which contain the object name, images, keypoints, and object descriptors from various positions and distances. All data were used in this paper is a new data that different from the previous research and the amount of each object is 50 pieces which mean more than previous research.

Figure 6 shows the flowchart of a multi-object recognition process using SIFT or SURF method that run sequentially on one thread only. The process to detect the number of keypoint from camera images and databases, the process of matching between descriptor of the camera images and database, determine the center of detected objects, and the process to calculate the keypoint and image descriptor of the camera are the realtime process of the matching data of all objects in the database using two *for* loops, first, for each file, and second, for each object in the file. In this research, "*batas_min_matching"* parameter of SIFT method is 70, and the SURF method is 25. The parameters is the detection accuracy of each method, so if the parameters are less than or equal to that value then one object can be detected more than once and the coordinates of the object become less precise.

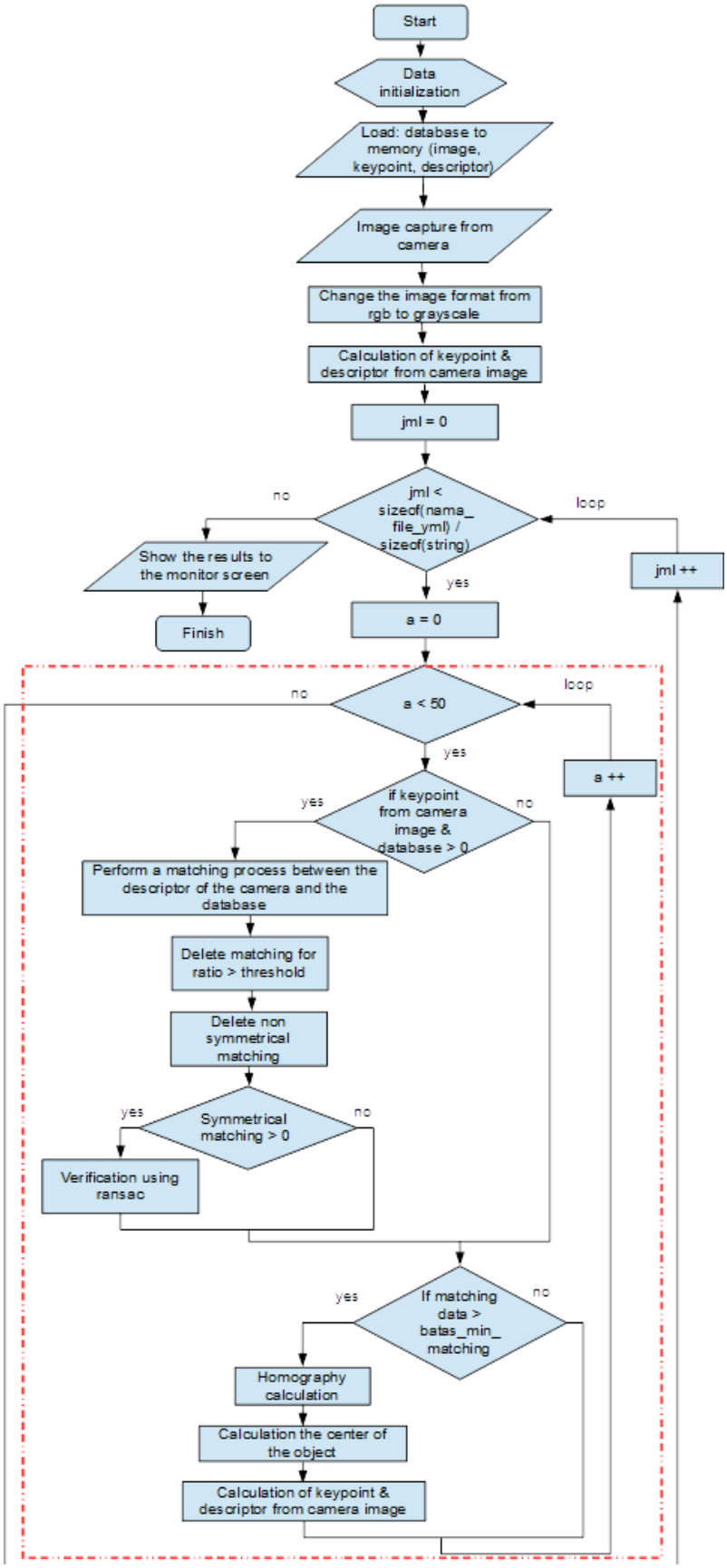


Figure 6. Flowchart of the multi-object recognition sequentially

The processing time of multi-object recognition sequentially using SIFT and SURF methods can be seen in Figure 7.

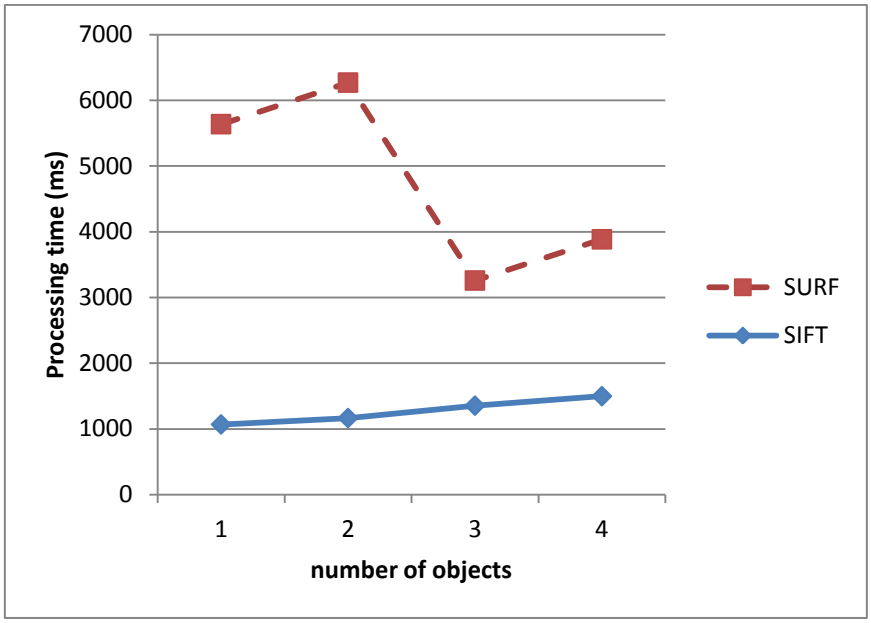


Figure 7. The processing time of multi-object recognition sequentially using SIFT and SURF methods

Figure 7 shows the average time graph of the multi-object recognition process sequentially using SIFT and SURF method to detect one to four objects at a time. The graph which is using red dashed line is SURF, and blue solid line is SIFT. The number of experiments in each object recognition process is 60 times, with ten different object positions and each object position is repeated six times. The processing time in recognizing one to four objects with SIFT method is 1067.35 ms, 1164.78 ms, 1352.93 ms, and 1497.35 ms, while the processing time in recognizing one to four objects by SURF method is 5635.99 ms, 6268.47 ms, 3256.63 ms, and 3883.78 ms. The processing time of object recognition using SIFT method is more linear than SURF method, it is related to several factors as the number of keypoints / descriptors, and the order of images in the database (beginning, middle or end of the database).

The loop of the multi-object recognition flowchart in Figure 6 will be processed in parallel using the openmp API version 1 that was default integrated in the visual studio. Openmp version 1 has the disadvantage, it is not able to execute more than one loop at a time, while the loop used in this paper is two *for* loop. Because of that, parallel processing is tried to be implemented in each loop. First, The parallel processing experiments have been performed on the first *for* loop as follows.

```
#pragma omp parallel for schedule(dynamic,1)
for (jml = 0; jml < sizeof(nama_file_yml) / sizeof(string); jml++) {
      for (a = 0; a < 50; a++) {
          /* The process of detecting the number of keypoint images
          from cameras and databases, the process matching between the
          descriptor of the camera images and database, determine the
          center of detected objects, until the process of calculating
          keypoint and image descriptor from camera */
      }
}
```

The experiment shows that the program can not be executed or an error occurs, it is related to a bug in version 1 openmp. Second, parallel processing experiments have been done on the second for loop which shows in a box with a red dashed line in Figure 6, the syntax can be seen as follows.

```
for (jml = 0; jml < sizeof(nama_file_yml) / sizeof(string); jml++) {
      #pragma omp parallel for schedule(dynamic,1)
      for (a = 0; a < 50; a++) {
          /* The process of detecting the number of keypoint images
          from cameras and databases, the process matching between the
          descriptor of the camera images and database, determine the
          center of detected objects, until the process of calculating
          keypoint and image descriptor from camera */
      }
}
```

The experiments show that the program can run well which the results can be seen in Chapter 3. Illustration of the parallel process on the *for* loop using the core i7-4790 processor can be seen in Figure 8.

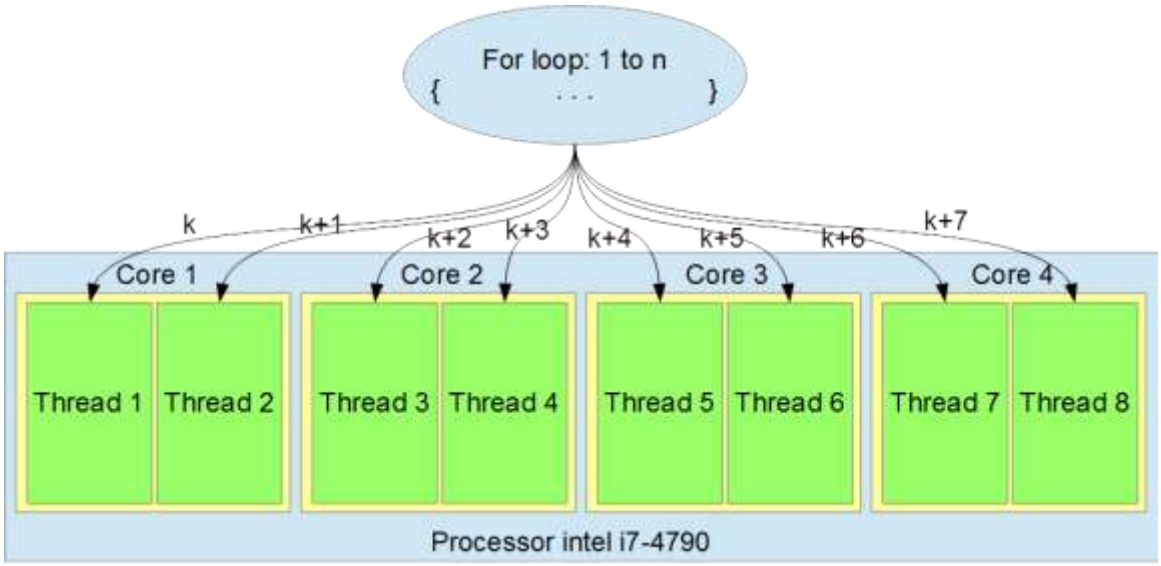


Figure 8. Illustration of the parallel process on the *for* loop using the core i7-4790 processor

Figure 8 shows an example of a parallel processing illustration of each *for* loop iteration on eight thread core i7-4790 processor which assuming that no other program is running on each thread. Experiments have also been performed on the *for* loop after optimization of the for loop syntax, which can be seen in Figure 9.

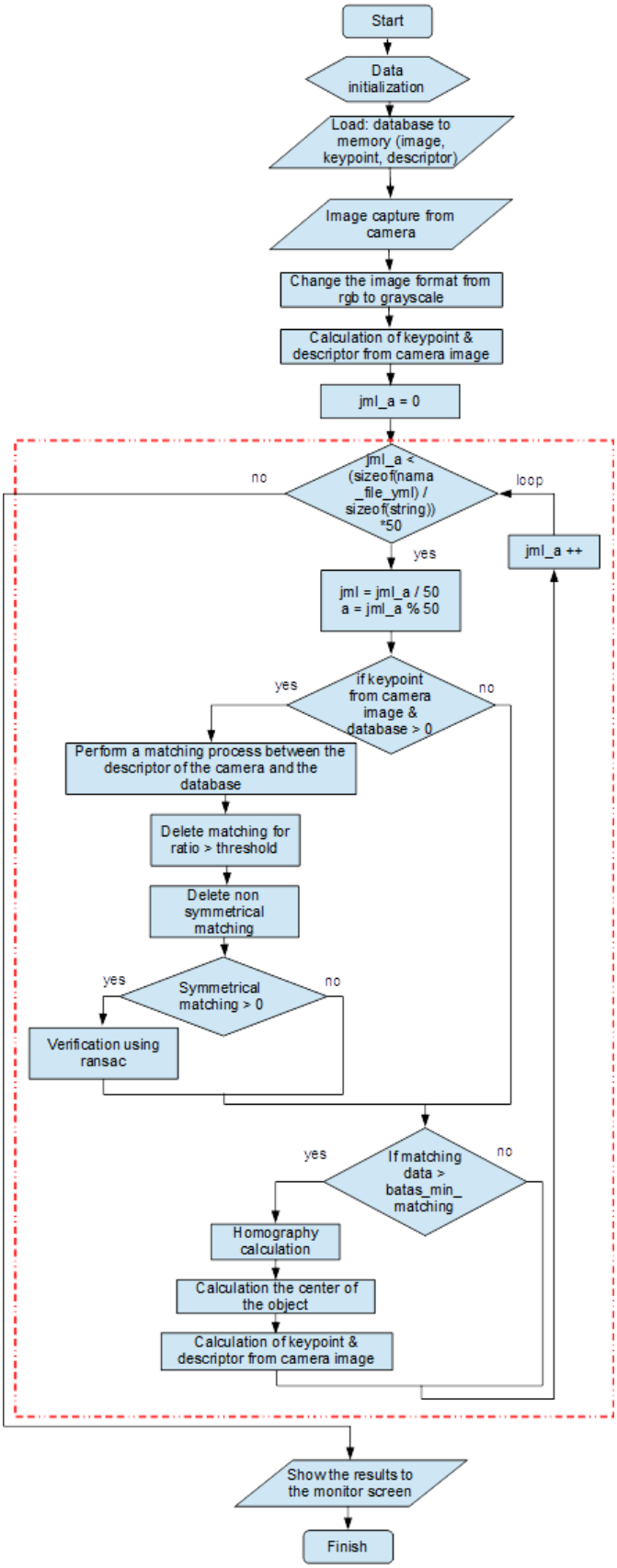


Figure 9. Flowchart of multi-object recognition after the for loop syntax have optimized

The box with the red dashed line in Figure 9 can be seen that both *for* loop were used are combined into one for, which means the number of for loop iterations is equal to the total number of data files and the contents of each data file. The program syntax can be seen as follows.

```
#pragma omp parallel for schedule(dynamic,1)
for (int jml_a = 0; jml_a < (sizeof(nama_file_yml) / sizeof(string)) *
50; jml_a++) {
      int jml = jml_a / 50;
      int a = jml_a % 50;
      /* The process of detecting the number of keypoint images from
      cameras and databases, the process matching between the
      descriptor of the camera images and database, determine the
      center of detected objects, until the process of calculating
      keypoint and image descriptor from camera */
}
```

jml is a variable that shows the index of an object file, and *a* is an index of the contents of each file. The processing time in this experiments is using *high_resolution_clock* library with *#include* <*chrono*> header [15].

3. Result and Discussion

When the experiments of the multi-object recognition software run, there are several other programs that also run on windows 8.1, i.e. *windows explorer*, *sticky notes*, *visual studio*, and *task manager*. The experiments have been done to see the CPU load before the multi-object recognition software runs, and when it runs both on sequential and parallel using the *task manager*. The display of CPU load was used before a multi-object recognition program runs can be seen in Figure 10.

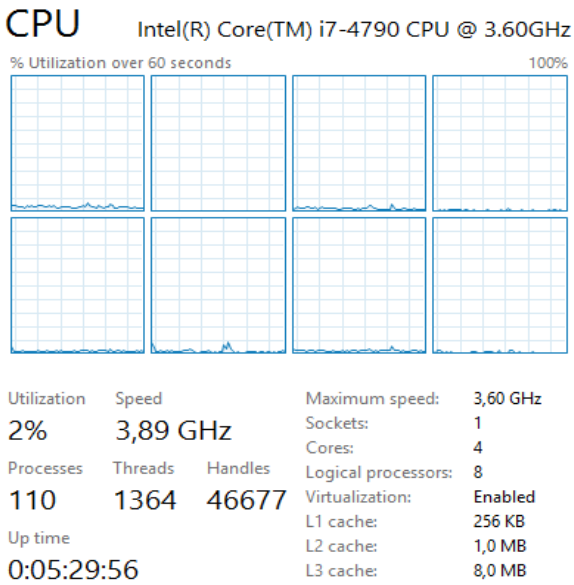
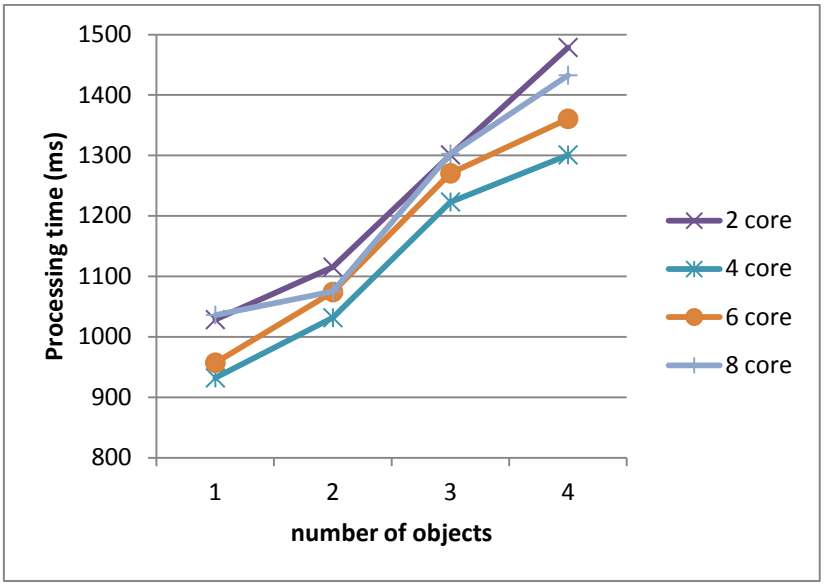


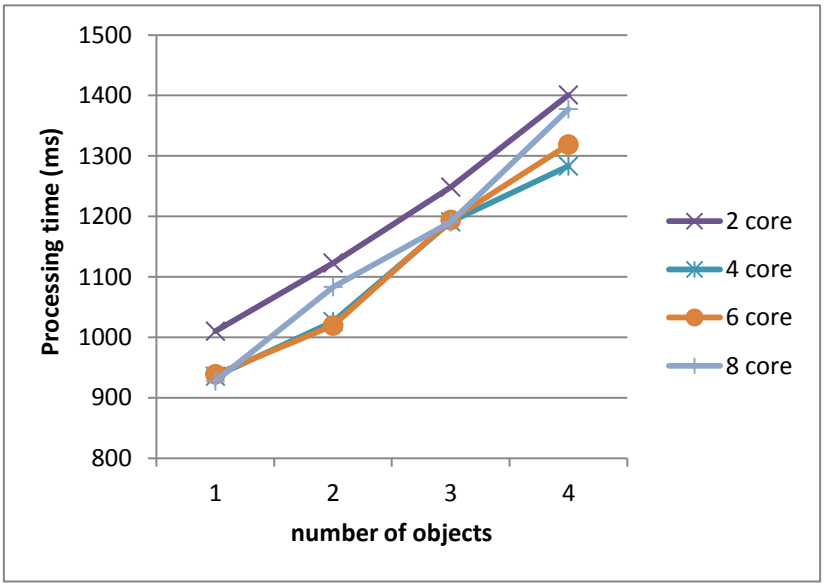
Figure 10. The CPU load before the multi-object recognition software running

Based on the experimental results, it can be seen that the CPU load before the multi-object recognition software runs is about 2%, when the multi-object recognition program runs sequentially is about 22%, and when the multi-object recognition program is running in parallel, the CPU load was used is greater than or equal to 66%, it means that the CPU load is greater than or equal to 44% compared sequentially.

The parallel processing time of multi-object recognition using SIFT method, with the original of the *for* loop syntax, and after optimization the of the *for* loop syntax can be seen in Figure 11.



(a)

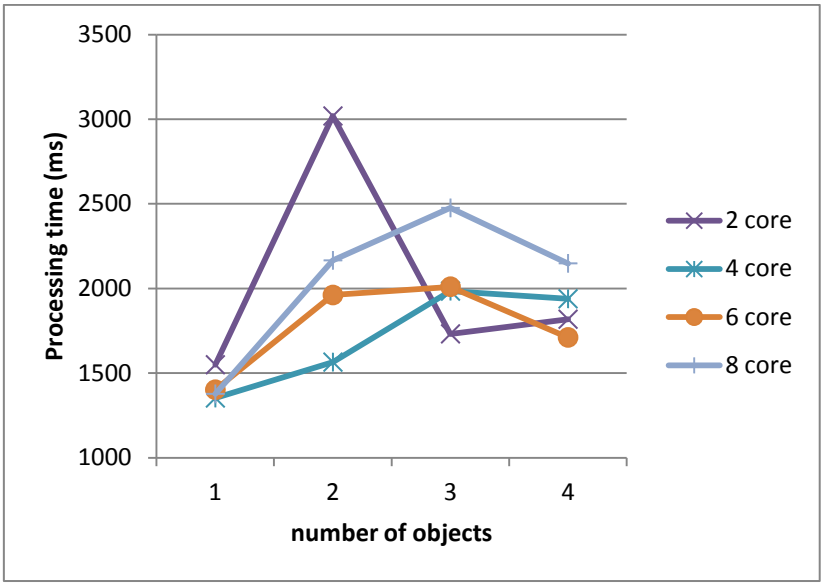


(b)

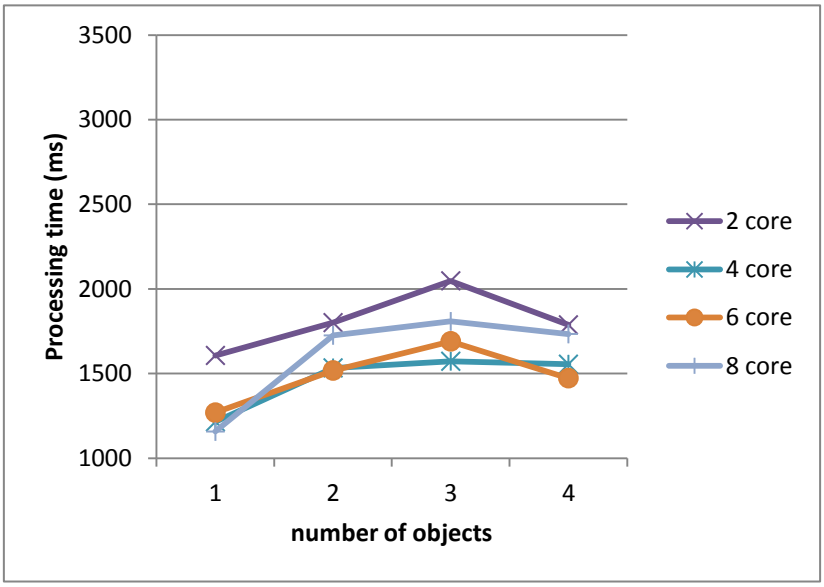
Figure 11. The parallel processing time of multi-object recognition using SIFT method using: (a) the original of the *for* loop syntax; (b) after optimization of the *for* loop syntax

Based on Figure 11, it shows that the parallel processing times of multi-object recognition using SIFT method in two, four, six, and eight cores is faster than sequentially, where the fastest time is obtained in the condition after optimization of the *for* loop syntax. The fastest time to recognize one to four objects is 927.13 ms (8 cores), 1019.31 ms (6 cores), 1190.72 ms (8 cores), and 1283.05 ms (4 cores), or it faster 13.14%, 12.49%, 11.99%, and 14.31% than sequential process.

Figure 12 shows the parallel processing time using the SURF method, with the original of the *for* loop syntax, and after optimization the of the *for* loop syntax.



(a)



(b)

Figure 12. The parallel processing time of Multi-object recognition time using SURF method, using: (a) the original of the *for* loop syntax; (b) after optimization the of the *for* loop syntax

Figure 12 shows the parallel processing time of multi-object recognition using SURF method using two, four, six, and eight cores is also faster than sequential, where the fastest time is obtained in the condition after optimization of the *for* loop syntax. The fastest time of recognizing one to four objects is 1157.13 ms (8 cores), 1517.83 ms (6 cores), 1572.14 ms (4 cores), and 1472.64 ms (6 cores), or it faster 79.47%, 75.79%, 51.73%, and 62.08% than sequential process.

The processing time after optimization of the *for* loop syntax is faster than the original of the *for* loop syntax because the loop process become optimal. On the original of for loop syntax, only

the second loop which is done in parallel, but after optimization of the *for* loop syntax, both of for loop are done in parallel.

4. Conclusion

The Parallel processing has been successfully implemented in multi-object recognition software on SIFT and SURF methods using openmp library in two conditions, first, on the original of the *for* loop syntax, and second, after optimization the of the *for* loop syntax. Based on the experiments, it is known that the processing time of recognition multi-object in parallel is faster than sequential process, where the fastest time is obtained in the condition after optimization of the *for* loop syntax on both SIFT and SURF method, with processing time in recognizing one to four objects on SIFT method is 927.13 ms (8 cores), 1019.31 ms (6 cores), 1190.72 ms (8 cores), and 1283.05 ms (4 cores), or it faster 13.14%, 12.49%, 11.99%, and 14.31% than sequential process, while the SURF method is 1157.13 ms (8 cores), 1517.83 ms (6 cores), 1572.14 ms (4 cores), and 1472.64 ms (6 cores), or it faster 79.47%, 75.79%, 51.73%, and 62.08% than sequential process.

Acknowledgments

The author would like to thank the Research Center for Electrical Power and Mechatronics - LIPI especially the Industrial Automation Research Group which has supported this research.

References

[1] D. Y. Park, “Direct calculation of inter-particle distance in suspension by image processing,” *Powder Technology*, vol. 330, pp. 252–258, May 2018.
[2] S. Saleem *et al.*, “Feature points for multisensor images,” *Computer & Electrical Engineering*, vol. 62, pp. 511–523, Aug. 2017.
[3] A. Husin *et al.*, “Poisonous Shrimp Detection System for Litopenaeus Vannamei using k-Nearest Neighbor Method,” *Lontar Komputer. Jurnal Ilmiah Teknologi Informasi*, vol. 9, no. 1, pp. 20–27, Apr. 2018.
[4] M. Mirdanies *et al.*, “Object Recognition System in Remote Controlled Weapon Station using SIFT and SURF Methods,” *Journal Mechatronics, Electrical Power, Vehicular Technology*, vol. 4, no. 2, p. 99, Dec. 2013.
[5] T. Sterling *et al.*, *High performance computing : modern systems and practices*. Cambridge: Morgan Kaufmann, 2018.
[6] B. Schmidt *et al.*, *Parallel programming : concepts and practice*. Cambridge: Morgan Kaufmann, 2017.
[7] S. E. Oh and J.-W. Hong, “Parallelization of a finite element Fortran code using OpenMP library,” *Advance in Engineering Software*, vol. 104, pp. 28–37, Feb. 2017.
[8] E. G. Pinho and F. H. de Carvalho, “An object-oriented parallel programming language for distributed-memory parallel computing platforms,” *Science of Computer Program.*, vol. 80, pp. 65–90, Feb. 2014.
[9] G. Oger *et al.*, “On distributed memory MPI-based parallelization of SPH codes in massive HPC context,” *Computer Physics Communication*, vol. 200, pp. 1–14, Mar. 2016.
[10] R. D. Phillips *et al.*, “An SMP soft classification algorithm for remote sensing,” *Computer & Geosciences*, vol. 68, pp. 73–80, Jul. 2014.
[11] A. Amritkar *et al.*, “Efficient parallel CFD-DEM simulations using OpenMP,” *Journal of Computational Physics.*, vol. 256, pp. 501–519, Jan. 2014.
[12] M. Mirdanies, “Optimization of Robot Telemonitoring System Software using multi-thread method,” *INKOM Jurnal*, vol. 11, no. 1, pp. 15–24, May 2018.
[13] Intel Corporation, “Intel® Core™ i7-4790 Processor (8M Cache, up to 4.00 GHz) Product Specifications.” [Online]. Available: https://ark.intel.com/products/80806/Intel-Core-i7-4790-Processor-8M-Cache-up-to-4_00-GHz. [Accessed: 07-May-2018].
[14] CPUID, “CPU-Z | Softwares | CPUID.” [Online]. Available: https://www.cpuid.com/softwares/cpu-z.html. [Accessed: 07-May-2018].
[15] cppreference.com, “std::clock - cppreference.com.” [Online]. Available: http://en.cppreference.com/w/cpp/chrono/c/clock. [Accessed: 07-May-2018].